Cybersecurity

RESEARCH Open Access

LSGAN-AT: enhancing malware detector robustness against adversarial examples

Jianhua Wang[1], Xiaolin Chang[1*], Yixiang Wang[1], Ricardo J. Rodríguez[2] and Jianan Zhang[1]

Abstract
Adversarial Malware Example (AME)-based adversarial training can effectively enhance the robustness of Machine Learning (ML)-based malware detectors against AME. AME quality is a key factor to the robustness enhancement. Generative Adversarial Network (GAN) is a kind of AME generation method, but the existing GAN-based AME generation methods have the issues of inadequate optimization, mode collapse and training instability. In this paper, we propose a novel approach (denote as LSGAN-AT) to enhance ML-based malware detector robustness against Adversarial Examples, which includes LSGAN module and AT module. LSGAN module can generate more effective and smoother AME by utilizing brand-new network structures and Least Square (LS) loss to optimize boundary samples. AT module makes adversarial training using AME generated by LSGAN to generate ML-based Robust Malware Detector (RMD). Extensive experiment results validate the better transferability of AME in terms of attacking 6 ML detectors and the RMD transferability in terms of resisting the MalGAN black-box attack. The results also verify the performance of the generated RMD in the recognition rate of AME.

Keywords: Adversarial malware example, Generative adversarial network, Machine learning, Malware detector, Transferability

Introduction

Malware is being regarded as a severe threat to cybersecurity (Zhou and Jiang 2012; Christodorescu et al. 2005). Up to 2020, more than 1.14 billion malware were reported and the amount of malware expected will reach 1244.47 million in 2021 ("Malware Statistics Trends Report | AV-TEST" 2021). As shown in Fig. 1, the histogram chart denotes the scale of malware, the line chart denotes the increment ratio of malware, the left y-axis denotes the amount of malware, the right y-axis is the increment ratio, and the x-axis denotes the year. Machine Learning (ML) detectors have been explored for malware detection, and have achieved preferable detection performance (Yuan et al. 2014; Lucas et al. 2021). However, their capability is challenged by adversarial attacks (Yuan et al. 2019; Huang et al. 2017), which are based on adversarial examples.

Adversarial examples, proposed by Szegedy et al. (2014), are generated by adding slight perturbations on original data and utilized to carry out adversarial attacks and implement Adversarial Machine Learning (AML) for enhancing the defense performance of the system (Biggio and Roli 2018), especially in malware detection (Chen et al. 2018). In general, there are three major types of approaches of generating adversarial examples: gradient-based, optimization-based, and Generative Adversarial Networks (GAN)-based (Xiao et al. 2018). The first two types of approaches have three major issues: (1) need to access the white-box architecture and have the knowledge of model parameters all the time (Xiao et al. 2018), (2) their optimization process is slow and can only optimize perturbation for one specific instance each time (Xiao et al. 2018), (3) the low perception quality of the adversarial examples (Wang et al. 2020). Therefore, because GAN-based methods have advantages in

*Correspondence: xlchang@bjtu.edu.cn
[1] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, Beijing 100044, China
Full list of author information is available at the end of the article

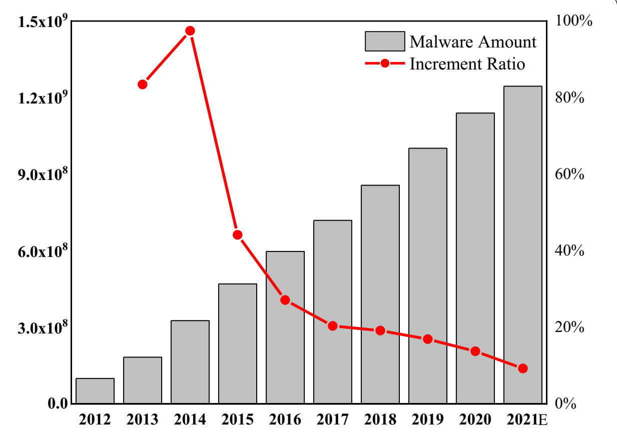

Fig. 1 The situation awareness of malware

obtaining the data distribution through unsupervised learning (Gonog and Zhou 2019), carrying out black-box attacks (Xiao et al. 2018), and generating more realistic/actual data (Gonog and Zhou 2019), the GAN-based methods are preferable for generating adversarial examples. However, the existing GAN-based Adversarial Malware Example (AME) generation methods need to be improved in several aspects, such as producing the higher quality AME by optimizing the boundary samples (Mao et al. 2017), resolving the problem of mode collapse (Creswell et al. 2018), and improving the stability of training (Creswell et al. 2018).

The above discussions motivate our work of this paper. We develop a novel LSGAN-AT for enhancing the robustness of malware detectors, which contains two main components including the LSGAN module and AT module.

The LSGAN module is a GAN-based AME generative model using least square loss function, which effectively tackles the aforementioned issues with the traditional GAN, and then generates smoother and high quality adversarial examples. Specifically, we utilize a generator, a discriminator and a trained well union detector with new structures. The generator combines perturbance and malware, and discriminator obtains labeled data from the union detector to update parameter gradients. After continuous training, LSGAN module outputs more effective AME.

The other component is the (Adversarial Training) AT module, which consists of a Feed-Forward Neural Network (FFNN)-based ML detector with new structure. Trained by original dataset and AME generated by the proposed LSGAN module, the AT module generates a Robust Malware Detector (RMD) which achieves a preferable recognition rate for AME and malware. Moreover, we conduct vast experiments to compare the defense performance using AME trained by different generative models.

Our contributions are following:

- We propose a new method, LSGAN module, to generate smoother and high quality adversarial examples. Two key features contribute to LSGAN's better performance, the Least Square (LS) loss function and a new network structure. LSGAN benefits more from its new network structure than the LS loss function to produce better performance. To the best of our knowledge, it is the first time to use the LS loss function to solve issues of GAN and design a GAN-based AME generative model.
- We propose a novel LSGAN-AT approach to produce a RMD against the black-box adversarial attacks effectively by adversarial training. Meanwhile, our AME and RMD show commendable transferability facing several ML-based detectors and different AME generative models respectively.

We use dynamic semantic feature Application Programming Interface (API) calls, which are extracted by running a virtual sandbox to preprocess raw dataset, to carry out extensive fine-grained experiments to evaluate our designs. Our experimental results indicate that:

- Compared with the MalGAN (Hu and Tan 2017), LSGAN can generate more effective adversarial examples to carry out adversarial attacks for a variety of malware detectors. Being attacked by adversarial examples generated by LSGAN, the recognition rate of detectors shows dramatically drop as shown in Table 6. Take the Multi-Layer Perceptron (MLP) as examples, the recognition rate is reduced from 97.81% to 1.09%. Extensive experiments reveal the effectiveness of our LSGAN.
- After being trained by MLP, the AME effective rate is more than 82.78% in 4 detectors (AdaBoost (AB), Logistic Regression (LR), Gradient Boosting decision tree (GB), Support Vector Machine (SVM)) and more than 69.40% in 6 detectors (AB, K-Nearest Neighbor (KNN), Decision Tree (DT), LR, GB, and SVM) as shown in Table 11. It means that AME generated by LSGAN has good attack transferability to black-box ML-based detectors.
- After being trained with the effective AME, RMD accomplishes effective defense against more than 65.33% AME. Moreover, after fine-grained comparison experiments, RMD achieves higher AME recognition rate than detector trained by AME generated by MalGAN as shown in Table 14.

The rest of this paper is organized as follows. "Related work" section is the discussions of related works. We describe the LSGAN-AT and its components as well as the training steps in 'LSGAN-AT approach". The numerical results and the experimental results are presented in "Experimental analysis and validation" section. "Conclusion" section provides conclusions.

Related work

This section introduces the related work of adversarial machine learning, the methods of adversarial attacks, and adversarial defenses.

Adversarial machine learning

AML is proposed to resist adversarial examples generated by adversarial opponents (Huang et al. 2011) and then protect the ML systems from threatens. There are several crucial strategies in AML including adversary introduction, existing adversarial attacks, and how to protect models (Biggio and Roli 2018). In this subtitle, we summarize the related work in two aspects: existing attacks and corresponding defense mechanisms.

According to the attacking occurrence phase, adversarial attacks can be classified into predicting/testing phase attack and training phase attack (Biggio and Roli 2018; Wang et al. 2019a). The former type is evasion attack, in which the original data input slight perturbations to evade a trained classifier at test time such as modifying the label to misclassify the test dataset without altering the decision boundary (as shown in left part of Fig. 2). The latter type is poisoning attack, in which poisoning instances contaminate the training dataset by injecting a small fraction of perturbation (Biggio and Roli 2018) with altering the model and the decision boundary (as shown in right part of Fig. 2).

According to adversarial attack types, there are two main defense mechanisms including proactive defense such as GAN-based method, and reactive defense such as data compression (Biggio and Roli 2018; Wang et al. 2019a). Reactive defense mechanisms update ML detectors by suffering some attacks while proactive defense mechanisms analyze possible vulnerabilities, flaws, and threatens to develop an ML detector.

In this paper, utilizing the LSGAN module, we implement an evasion attack by generating AME to avoid ML malware detection. Furthermore, we conduct a proactive defense mechanism method trained by AME for obtaining robust malware detectors.

Adversarial malware examples for evasion attack

The evasion attack has extensive applications using adversarial examples (Suciu et al. 2019). Li and Li (2020) proposed a mixture of attacks to generate AME without ruining its malicious functionality using semantic characteristics and byte features. Grosse et al. (2016) constructed highly effective AME crafting attacks using an improved FFNN. Chen et al. (2017) proposed an evasion attack by misleading malware being classified as benign. The other research of Grosse et al. (2017) expanded the original method (Papernot et al. 2016) to handle binary malware features without discarding the functionality. Khoda et al. (2019) proposed 2 novel approaches including a probability measure kernel-based and distance-based for selecting AME. Wang et al. (2019b) crafted AME using JSMA (Jacobian-based Saliency Method Attack) and trained Neural Network (NN). However, most of them utilized static features to construct AME and are vulnerable to code obfuscation techniques. Further, it is difficult to evaluate the AME effectiveness without enough AME evaluation.

GAN-based approach as a type of generating adversarial example (Xiao et al. 2019). Researchers have developed a GAN-based approach to deploy an evasion attack in the malware detection field. Hu and Tan (2017) proposed MalGAN, using a substitute detector as a basic malware detector and the basic GAN method in training,

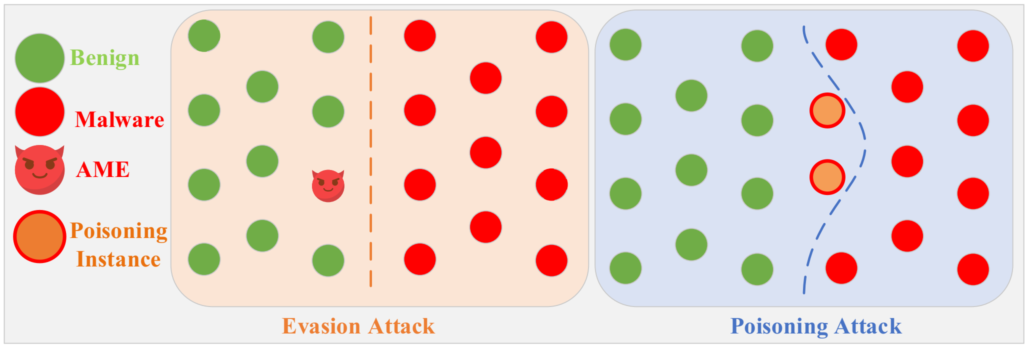

Fig. 2 Illustration of evasion attack and poisoning attack

to generate adversarial malware examples for attacking ML detectors. Yuan et al. (2020) inputs discrete malware binaries to continuous space, then feeds them to the generator of GAPGAN to generate adversarial payloads. However, the basic GAN exposes several problems, such as instability, gradient disappearance, and model collapse in the training phase (Lugmayr et al. 2020).

In this paper, we focus on the dynamic features such as API features extracted by using sandbox and improve the MalGAN using the least square loss function and the brand-new network of generator and discriminator. After being compared by experiments, the LSGAN module can obtain more effective adversarial malware examples and generating AME has better transferability. Table 1 compares these adversarial attack works in malware detection from the aspects of four main metrics, namely dynamic feature, model comparison, model transferability, and AME evaluation.

Adversarial malware examples for proactive defense

GAN-based methods, as a useful and efficient proactive defense approach, are frequently used for enhancing the defense performance of ML detectors or classifiers using AML. Recently, Zhang et al. (2021) proposed to add perturbations noise in bit sequences to generated adversarial examples and then enhanced the classification ability of Deep Neural Network (DNN) through adversarial training. However, bit sequences are malware static features, so a portion of malware can escape the detection easily using obfuscated code techniques. Li and Li (2020) proposed mixture attacks by using multiple generative methods to generate AME and enhance the defense by training an ensemble DNN model. Grosse et al. (2016) proposed an improved FFNN to generate AME and reduced the sensitivity of networks for defense. Chen et al. (2017) exploited the EvnAttack model to retrain the classifier progressively and apply evasion cost to regularize the optimization problem. Grosse et al. (2017) proposed an improved attack model for generating adversarial examples and investigated defense mechanisms for malware detection models trained using DNNs. Khoda et al. (2019) proposed two novel approaches, including a probability measure kernel-based method and distance-based method from malware cluster center for selecting AME and for retraining a classifier to defense the

Table 1 Related work of evasion attacks in malware detection

Ref	Attack phase	Data type	Data level	Model	Dynamic Feature	Model Comparison	Model transferability	AME evaluation
Hu and Tan 2017)	Training	Malware by Website (malwr.com)	SC	GAN	●	○	○	●
Li and Li 2020)	Training	Drebin(Arp et al. 2014), Androzoo (Allix et al. 2016)	SC, Byte	Ensemble Methods (DL-based)	○	●	●	○
Grosse et al. 2016)	Training	Drebin	SC	FFNN	○	●	●	○
Chen et al. 2017)	Testing	Windows	SC	EvnAttack (ML-based)	○	●	○	○
Grosse et al. 2017)	Training	Drebin	SC	DNN	○	●	○	●
Khoda et al. 2019)	Training	Drebin	SC	Ensemble Methods (ML-based)	○	●	●	○
Wang et al. 2019b)	Training	Drebin	SC	JSMA and NN	○	●	●	●
Yuan et al. 2020)	Training	VirusTotal ("VirusTotal". 2021), Kaggle 2015 ("Microsoft Malware Classification Challenge (BIG 2015), Chocolatey ("Chocolatey—The package manager for Windows" 2021)	Byte	GAPGAN	○	●	●	●
Ours	Training	VirusShare ("VirusShare.com". 2021), Androzoo	SC	LSGAN	●	●	●	●

[a] *SC* Semantic characteristic

[b] The Attack Phase includes during the training phase and during predicting/testing. The Data Type indicates the type of malware datasets such as Android and Windows malware. The Data Level displays the feature type of data, mainly including semantic characteristic, byte, and pixel. The Model demonstrates the trained model pattern including DL-based models and conventional ML-based models. The Approach exposes the overview of the specific method. The Dynamic Feature is used to analyze whether the paper uses dynamic malware features to avoid code obfuscation. The Model Comparison represents whether the paper makes the comparison. The Model Transferability indicates whether the paper attack other detectors using AME generated by themselves. The AME Evaluation denotes whether the reference paper makes the adversarial malware example performance analysis

attack. Sewak et al. (2020) developed the DOOM system using the Opcode feature based on deep reinforcement learning to enhance the intrusion detection system defense mechanism. Wang et al. (2017) proposed a new adversary resistant technique that obstructs attackers from constructing impactful AME by randomly nullifying features within samples. However, most the proactive defense methods lack components analysis and AME evaluation, so that researchers cannot obtain the adequate robustness of the model.

In this paper, we propose a proactive RMD, which is trained by AT module with AME generated by the LSGAN module. Compared to other robust malware detectors, we achieve integrated model analysis and evaluation using the whole metrics. In the experimental section, we conduct abundant experiments to demonstrate the effectiveness of our RMD. Table 2 shows the comparison of our work with the above defense models, we introduce the related work of adversarial defense in malware detection using four main metrics, namely, robustness analysis, component analysis, AME generation, and AME evaluation.

LSGAN-AT approach

Figure 3 shows the LSGAN-AT structure, composed of LSGAN and AT modules. The first module consists of a Generator (**G**), a Discriminator (**D**), and a Union Detector (**UD**). The training process with ML-based detector (**MD**) makes up the AT module. There are three types of inputs to LSGAN-AT: Malware, Perturbance and Benign software (Benign). The output of LSGAN-AT is robust **MD** (RMD), which distinguishes the malware and AME. **Algorithm 1** shows the LSGAN-AT training details. Note that we set epoch as iteration times. LSGAN and AT modules are detailed in "LSGAN module" and "AT module" sections, respectively.

Table 2 Comparison of the existing adversarial defense approaches in malware detection

Ref	Defense mechanism	Data type	Data level	Model	Robustnes analysis	Components analysis	AME generation	AME evaluation
Li and Li 2020)	Proactive	Drebin, Androzoo	SC, Byte	Ensemble Methods (DL-based)	●	○	●	●
Grosse et al. 2016)	Proactive	Drebin	SC	FFNN	●	●	●	○
Chen et al. 2017)	Reactive	Windows	SC	SecDefender (ML-based)	○	●	○	○
Grosse et al. 2017)	Proactive	Drebin	SC	DNN	○	○	●	●
Khoda et al. 2019)	Proactive	Drebin	SC	Ensemble Methods (ML-based)	●	○	●	○
Sewak et al. 2020)	Proactive	MALICIA (Nappa et al. 2015)	Opcode	Deep reinforcement learning	○	○	●	○
Wang, et al. 2017)	Reactive	Window Audit Log (Berlin et al. 2015), MINST, CIFAR-10	SC, Pixel (image)	DNN	●	○	●	●
Ours	Proactive	VirusShare, Androzoo	SC	LSGAN	●	●	●	●

[a] The Defense Mechanism includes proactive and reactive. The Robustness Analysis aims to evaluate whether the defense model can recognize the AME of other attack models. The Component Analysis is used to evaluate whether the reference paper makes components analysis. The AME generation exposes whether the defense model can generate AME, and normally, a proactive defense mechanism will generate the AME

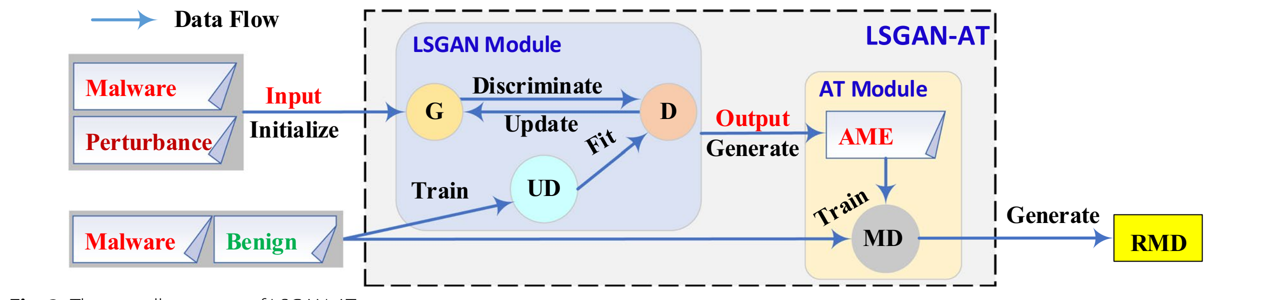

Fig. 3 The overall structure of LSGAN-AT

```
Algorithm 1 The LSGAN-AT Training
Input: Benign $B$, Malware $M$, Perturbance $z$
Output: $ACC$, $REC_{adv}$ and $REC_{mal}$
Initialize: parameter $\theta_g$, $\theta_d$
1:  /* The LSGAN Module */
2:  Randomize samples $B$ and $M$
3:  Train UD using $B$ and $M$ with ML-based detector
4:  Label $M$ and $B$ using UD {M:1, B:0}
5:  Fit UD strategy to D
6:  For epoch in the range:
7:      Generate adversarial malware examples $M' = G_{\theta_g}(M, z)$
8:      minimize G loss $L(G)$ and D loss $L(D)$
9:      Update parameter $\theta_g$ by $\nabla\theta_g L(D) \rightarrow \theta_g$
10:     Update parameter $\theta_d$ by $\nabla\theta_d L(D) \rightarrow \theta_d$
11: End for
12: /* The AT Module */
13: Input $B$, $M$ and $M'$
14: Train MD using $B$, $M$ and $M'$ with new FFNN-based structure
15: Classify $M$ and $M'$ using MD
16: Update $ACC$, $REC_{adv}$ and $REC_{mal}$
```

LSGAN module

LSGAN firstly initializes the generator **G** and produces AME combining malware and several subtle perturbances, and use **UD** to fit a proven detector by vast experiments. Then the generator **D** is utilized to distinguish the AME and update the generation strategy of **G**. This constitutes an LSGAN iteration (Lines 2–11). During the training, a phased optimization strategy is used, and both **G** and **D** are alternately optimized to reach equilibrium. After **G** and **D** reaching Nash equilibrium (Nash 2016), we obtain AME generated by LSGAN. The working processes of **G** and **D** are given as follows. Their structures shown in Table 3. To help the **D** to classify malware efficiently, we add a **UD** using Multi-Layer Perception (MLP) structure. Note that we will demonstrate the selection standard and experimental results in detail in "Parameter and component selection strategy" section.

To be specific, firstly, we utilize the **G** to generate AME by combining a slight perturbance with malware. Then, we use the **UD** to fit a proven detector and obtain the data label to the **D**. The **D** is used to distinguish the AME, and it update the generation strategy of **G** for producing effective AME to avoid the detection of **D**.

G aims to transform the feature vector into an adversarial example. To be specific, **G** takes an M-dimensional malware x and a Z-dimensional vector slightly perturbance z as input. Here, z is a random vector that follows uniform distribution in range $[0, 1)$. We adopt batch normalization to make sure that the input of each layer has a mean value of 0 and a variance of 1. In addition, Rectified Linear Units (ReLU) (Nair and Hinton 2010) will set all negative gradients to zero and lose some gradient information. To overcome the information loss, we utilize LeakyReLU (Chen et al. 2018) with a confirmed coefficient a_i= 0.2 as in Eq. (1). Furthermore, the **G** uses Mean-Square Error (MSE) as a loss function and Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.0002.

$$f(x) = \begin{cases} x_i & x_i > 0 \\ a_i x_i & a_i \neq 0, x_i \leq 0 \end{cases} \quad (1)$$

Table 3 The network structure of the generator and the discriminator in LSGAN

The generator (G) network structure		The discriminator (D) network structure	
Layer type	**Output shape**	**Layer type**	**Output shape**
Input Layer (Malware)	(None, 128)	Input Layer	(None, 128)
Input Layer (Noise)	(None,20)	LeakyReLU(Dense)	(None,256)
Concatenate (Malware + Noise)	(None,148)	Dropout(0.05)	(None,256)
LeakyReLU (Dense)	(None,256)	LeakyReLU(Dense)	(None,256)
Batch Normalization	(None,256)	Dropout(0.05)	(None,256)
LeakyReLU(Dense)	(None,256)	LeakyReLU(Dense)	(None,256)
Batch Normalization	(None,256)	Dropout(0.05)	(None,256)
LeakyReLU (Dense)	(None,256)	Sigmoid(Dense)	(None,1)
Batch Normalization	(None, 256)		
Sigmoid (Dense)	(None,128)		
Maximum	(None,128)		

The purpose of **D** is to learn the boundary of malware and benign applications. To be specific, the **D** uses an M-dimensional vector as input. After multi-layer hidden layers, the **D** uses the Sigmoid activation function to output the predicted probability of malware. In addition to the output layer, other layers use the LeakyReLU activation function. Furthermore, we utilize the dropout layer to prevent **D** from overfitting. Same as the **G**, the discriminator uses MSE as loss function and Adam optimizer with a learning rate of 0.0002.

Note that the goal of GANs is to generate examples with the close probability distribution of datasets. By the means of many times iterations, GANs reduce the distance between the samples distribution and real samples distribution. There are three methods of adversarial loss function optimization-based GANs, including f-Divergence-based methods, Integral Probability Metrics (IPMs)-based methods, and other loss function methods (Pan et al. 2020). The Jenson-Shannon divergence, as one of f-Divergence-based methods, used by regular GAN and MalGAN (GAN in malware dataset), has the problem of unstable training and gradient vanishing (Creswell et al. 2018). The IPMs-based objective functions such as WGAN (Arjovsky et al. 2017) using Wasserstein distance, have been verified in solving the problems of regular GAN. However, the weight clipping could not converge and may reduce the quality of generated examples (Pan et al. 2020). In addition, the other loss function such as Energy-based GAN (EBGAN) (Zhao et al. 2017), they replaces the discriminator with an energy function and effectively increases the diversity of generated examples. However, the result of extensive experiments does not perform satisfactorily (Pan et al. 2020).

As a result, to tackle the problems of regular GAN, Least-Square GAN (Mao et al. 2017) as another f-Divergence-based method, penalizes samples which are judged correctly but far from the decision boundary. This approach makes the generated samples more realistic (Pan et al. 2020). Meanwhile, in the Computer Vision field like facing discrete pixel data, Least-Square GAN has a realistic performance comparing the other 13 GAN models (Pan et al. 2020). Similarly, the malware dataset has the same discrete feature data, thus we believe the LSGAN will perform better than others. Meanwhile, we conduct extensive experiments to validate the effectiveness of LSGAN in "Experimental analysis and validation" section.

$$L(G) = \frac{1}{2}\mathbb{E}_{x\sim M,z}\left[\left(D_{\theta_d}\left(G_{\theta_g}(x,z)\right)\right)^2\right] \tag{2}$$

Given that the label of malware is 1 and the label of benign is 0, the loss function of the improved **G** is shown in Eq. (2). M is a malware set and $G_{\theta_g}(x,z)$ denotes adversarial examples generated by **G**. $D_{\theta_g}\left(G_{\theta_g}(x,z)\right)$ denotes the probability value of adversarial examples discriminated by **D**. We minimize **G** loss $L(G)$ to reduce $D_{\theta_g}\left(G_{\theta_g}(x,z)\right)$. The loss function of the improved discriminator **D** as Eq. (3). UD_{Benign} is a benign set and

Table 4 The malware detector (MD) structure in AT

Layer type	Output shape
Input Layer	(None,128)
LeakyReLU(Dense)	(None,256)
Dropout(0.05)	(None,256)
LeakyReLU(Dense)	(None,128)
Dropout(0.05)	(None,128)
LeakyReLU(Dense)	(None,256)
Dropout(0.05)	(None,256)
Sigmoid(Dense)	(None,1)

$UD_{Malware}$ is a malware set classified by **UD**. We minimize D-loss $L(D)$ to distinguish between benign and malware.

$$L(D) = \frac{1}{2}\mathbb{E}_{x\sim\, UD_{Benign}}\left[\left(D_{\theta_d}(x)\right)^2\right] + \frac{1}{2}\mathbb{E}_{x\sim\, UD_{Malware}}\left[\left(D_{\theta_d}(x)-1\right)^2\right] \tag{3}$$

AT module

The **MD** is used to get the prediction of the malware (including AME and malware) and the benign applications, and utilizes an FFNN as a basic network structure. In addition to the output layer, **MD** uses the ReLU as the activation function and uses dropout (0.05) to avoid overfitting. In the output layer, the Sigmoid, as the activation function, is used to output the probability of AME. The network structure of the **MD** is shown in Table 4. Furthermore, we use adversarial dataset to train AT module with **MD**. At last, we obtain a trained well RMD.

Experimental analysis and validation

This section shows our experimental evaluation results from the aspects of data preprocessing, evaluation metrics, LSGAN-AT components evaluation, component selection strategy, the transferability of LSGAN-AT components, and experimental comparison. As investigated in Damodaran et al. 2017), researchers conducted experiments in malware detection with three approaches including static data approaches, dynamic data approaches, and hybrid approaches, and found a fully dynamic approach based on API calls was extremely effective across a range of malware families. Hence, according to the investigation of Table 1, we select MalGAN as the main comparison method in adversarial attack and adversarial defense. The AT module of MalGAN is implemented by training with the AME generated by MalGAN.

Data preprocessing

The malware dataset (M) used in our experiments is got from VirusShare ("VirusShare.com". 2021), and the benign set (B) is obtained from AndroZoo (Allix et al. 2016). A total of 3733 malware and 2357 benign are collected. We use Cuckoo sandbox ("Cuckoo Sandbox—Automated Malware Analysis". 2021) to virtually run the malware and benign and to extract API calls of each M and B, and then generate a vector $\Phi(x)$ of M and B using Eq. (4). Here, v denotes a factor of API features V. Then, we rank API calls according to the API importance using 'feature_importances_' attribute in Scikit-Learn library Random Forest classifier with 2000 estimators, 0 random state and -1 jobs. After that, we utilize One-Hot method to obtain the 128 most important and frequently-used API features as 128-dimension vector. Note that the value of each dimension represents the corresponding API, namely '1' for existing API in certain malware/benign and '0' for not. We will demonstrate the selection strategy of API dimension in "Parameter and component selection strategy" section.

Table 5 Performance of detectors (%)

Component	train set		Test set	
	REC	ACC	REC	ACC
UD	98.17	96.54	97.81	96.69

20% data is for the test, 40% data is for training the LSGAN module and 40% data is for training the AT module. Adv is defined to denote the Adversarial dataset (Adv), which is composed of malware, AME, and benign applications. Ori is defined to denote the Original dataset (Ori), which is the set of malware and benign.

$$\Phi : X \to \{0,1\}^{|v|}, \Phi(x) \to (I(x,v))_{v\in V} \tag{4}$$

Evaluation metrics

Five metrics are used to evaluate the performance of the model and detectors: ACCuracy (ACC), Adversarial example Effectiveness Rate (AER), RECognition rate (REC), True Positive Rate (TPR), and False Positive Rate (FPR). Given that True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), Eq. (5) defines ACC and Eq. (6) defines TPR and FPR.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \tag{5}$$

$$TPR = \frac{TP}{TP+FN} \times 100\%, \quad FPR = \frac{FP}{TN+FP} \times 100\% \tag{6}$$

REC evaluates the AME recognition performance of detector and defense model. Compared with ACC, REC represents the precision which measures the closeness of each AME and effective AME should have high ACC and high REC. Note that in order not to confuse ACC with precision, we use REC to present the detect precision of detector and defense model. AER is defined to evaluate the effectiveness of AME. If most adversarial examples can avoid **MD** detection, this LSGAN module is thought as effective. The calculation formulas of AER and REC are shown in Eq. (7).

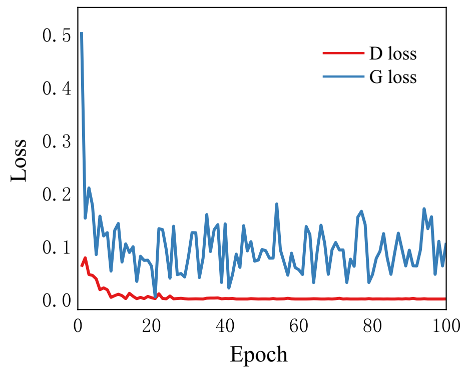

Fig. 4 Training loss of discriminator and generator

$$REC = \frac{n_{recognizedAME}}{N_{allAME}} \times 100\% = \frac{TP}{TP + FP},$$
$$AER = \frac{n_{EffectiveAME}}{N_{allAME}} \times 100\% = 1 - REC \qquad (7)$$

LSGAN-AT evaluation

We first evaluate **UD** and LSGAN performance, and then evaluate LSGAN-AT.

1. UD performance

UD is a crucial component in LSGAN-AT. Only if it can achieve effective detection, it can be used as the union training detector in the LSGAN module. In other words, better performance in malware detection will greatly enhance the learning ability of LSGAN module. As shown in Table 5, REC of **UD** is 97.81% and the ACC of **UD** is 96.69%. Obviously, **UD** has good malware detection performance.

2. LSGAN performance

LSGAN module is the most important part of the LSGAN-AT. It is evaluated from the aspects of training loss and the effectiveness of AME under various detectors. We utilize D loss as the discriminator training loss and G loss as the generator training loss. Figure 4 shows that after 20 epochs, G loss reaches convergence and D loss varies within a small range. It means that LSGAN has a preferable convergence ability.

To testify the performance of LSGAN in AME generation, 7 basic ML malware detectors are tested respectively and then their experimental results are compared in Ori and Adv. These 7 basic MDs include MLP (with 50 hidden layer sizes, SGD solver, and 0.1 learning rate), Decision Tree (DT), AdaBoost (AB), Logistic Regression (LR, with L2 regularization), Gradient Boosting decision tree (GB), K-Nearest Neighbor (KNN, with k=6), and Support Vector Machine (SVM). They are from Scikit-Learn Library (Pedregosa et al. 2011) in Python. Table 6 shows results. Under MLP detector, the REC of Adv is 1.09%, while the REC of Ori is 97.81% and the AER is 96.72%. In addition, the ACC of Ori is 96.14% while the ACC of Adv is 23.14%. We observe that LSGAN module is more effective than the other detectors. Especially, compared to DT, LR, MLP, and GB, AER of LSGAN is better by more than 96.35%.

3. LSGAN-AT performance

We aim to demonstrate the performance of RMD in terms of the REC rate. As shown in Table 7, with training deploying, the result of RMD is reaching 65.33%. That

Table 6 The performance of the LSGAN module (%)

Detector	REC		AER
	Ori	Adv	
AB	97.08	7.66	89.42
DT	100.00	0.00	100.00
LR	98.54	2.19	96.35
MLP	97.81	1.09	96.72
SVM	98.91	28.47	70.44
KNN	97.45	20.07	77.38
GB	99.64	2.55	97.09

Table 7 The AME recognition Rate OF RMD (%)

Component	Train Adv REC	Test Adv REC	Ori-Mal REC
RMD	65.90	65.33	97.03

Table 8 The top ten API and its importance

API	Importance
NtTerminateProcess	0.049898
GetFileType	0.037770
RtlAddVectoredExceptionHandler	0.037261
RegOpenKeyExA	0.035865
NtCreateThreadEx	0.021470
LoadResource	0.020279
RegQueryValueExA	0.018552
NtProtectVirtualMemory	0.018494
WriteConsoleA	0.018434
NtOpenSection	0.018234

Table 9 The time consumption and REC of different API dimension

API dimension	Time Consumption (s)	REC (%)
32	43.13	39.05
64	45.57	40.88
128	48.46	**65.33**
256	49.70	54.01
265	50.72	38.32

Table 11 The performance of adversarial examples in other detectors (%)

Detector	REC Ori	REC Adv	AER
AB	97.18	7.66	92.70
DT	98.18	20.51	79.67
LR	98.63	7.18	89.75
SVM	97.18	9.52	87.05
KNN	98.91	29.63	69.40
GB	86.64	15.35	82.78

means our LSGAN-AT can recognize the proportion of AME is 65.33% in Adv, and more than 97% of malware can be identified in Ori.

Parameter and component selection strategy

In this subsection, we will demonstrate the selection strategy of API dimension and UD component.

1. Why LSGAN is fed by 128 dimension input vector

As statements in subsection data preprocessing, we obtain 265 dynamic features APIs using Cuckoo sandbox and rank API importance using RF classifier. The top ten important APIs are shown as Table 8.

Based on the above APIs, we select different API dimensions such as 32, 64, 128, 256, and 265 to compare time consumption and REC as shown in Table 9. Meanwhile, the other crucial reason why we select 128 as API dimension is that, to control variable, we utilize the same API dimension with MalGAN (Xiao et al. 2019) to reveal the effectiveness of least square loss and our networks.

2. Why MLP could be used in UD

A better detector should have higher ACC, higher TPR, and lower FPR. Because deep learning classifiers are time-consuming, we select several commonly used traditional ML classifiers to serve as malware detectors. Table 10 shows the detection performance of different detectors. After comparing by experiments, we select the best detector MLP as the **UD**. Note that we execute independent experiments using malware dataset to select the best ML detector.

Table 10 The performance of malware detectors (%)

Detector	ACC	TPR	FPR
AB	95.04	99.27	17.98
DT	92.28	99.27	29.21
LR	95.31	99.27	16.85
MLP	96.48	99.63	13.48
SVM	92.84	99.27	26.97
KNN	94.49	98.17	16.85
GB	95.59	99.27	15.73

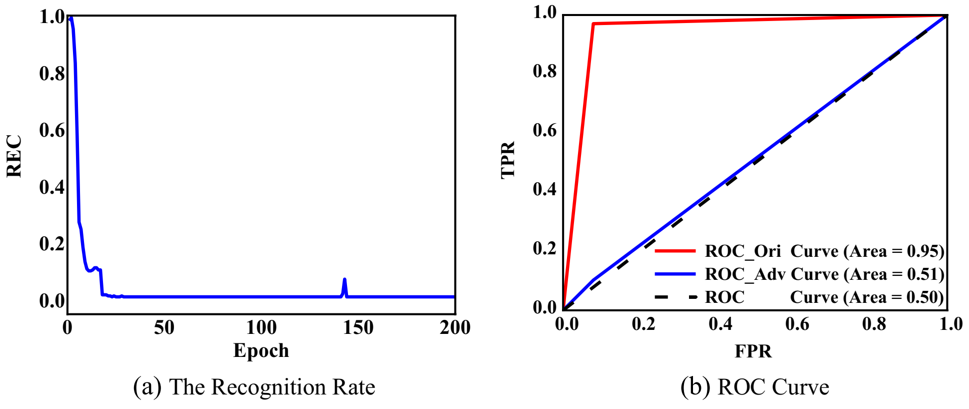

Fig. 5 LSGAN results in MLP detector

Table 12 The recognition rate of AME generated by MalGAN and LSGAN (%)

Component	REC		
	Train Adv	Test Adv	Ori-Mal
RMD (MalGAN)	56.49	52.20	99.64
RMD (LSGAN)	65.90	65.33	97.03

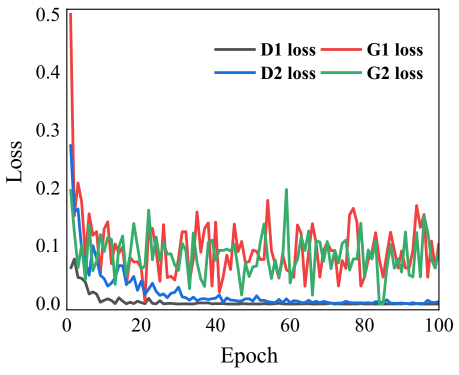

Fig. 6 The convergence of training loss

LSGAN-AT transferability

LSGAN-AT transferability is evaluated from the aspects of the AME transferability of the LSGAN module and the transferability of the RMD.

1. AME transferability of LSGAN module for attack

Figure 5 shows LSGAN results in the MLP detector. In Fig. 5a, after 200 epochs, the REC of the adversarial examples is lower than 0.1. And in Fig. 5b, the ROC (Receiver Operating Characteristic) area is lower than that of the original examples 44%. This means that the AME can effectively attack the detector. In this experiment, the AER of LSGAN using MLP detector is 98.65%.

To test the transferability of the adversarial examples, we select MLP as a detector and train the model to generate adversarial examples. The generated results are detected in SVM, LR, DT, AB, KNN, and GB to verify the transferability of adversarial examples. Table 11 shows the transferability. Except for DT and KNN detectors, adversarial examples generated by LSGAN and trained by MLP mostly achieve more than 82.78% AER. It may be due to that DT and KNN have a completely different structure from the neural networks, so their transferability is less than that of other detectors.

2. Transferability of RMD

An excellent transferability of RMD is crucial. The transferability means that our RMD can recognize the AME by other generative models. To be specific, we deploy MalGAN (Hu and Tan 2017) as an attack model and use RMD to identify the malware and AME generated by MalGAN. The experimental results are shown in Table 12. The RMD, proposed by this paper, achieves a preferable defense transferability. This RMD can recognize more than 50% of adversarial examples generated by MalGAN. It indicates our RMD is dependable under MalGAN adversarial attack to some extent.

Experimental comparison

We will use three subtitles to demonstrate the different models including the convergence of GAN training loss, different combinations of networks and loss functions in generating AME, and the comparison of REC.

1. The convergence of training loss fuction

We can discover that the LSGAN and the MalGAN have a preferable performance of training, as shown in Fig. 6.

Table 13 The recognition rate comparison of different combinations (%)

Detector	REC				
	Original Result	(LSGAN) $s1+f1$	(CE1) $s2+f1$	(CE2) $s1+f2$	(MalGAN) $s2+f2$
	Ori	Adv	Adv	Adv	Adv
AB	97.08	7.66	10.58	8.03	30.66
DT	100.00	0.00	10.22	0.00	12.41
LR	98.54	2.19	10.95	2.55	14.96
MLP	97.81	1.09	2.19	1.46	36.50
SVM	98.91	28.47	52.92	35.40	65.69
KNN	97.45	20.07	44.16	18.25	73.72
GB	99.64	2.55	22.26	2.19	10.22

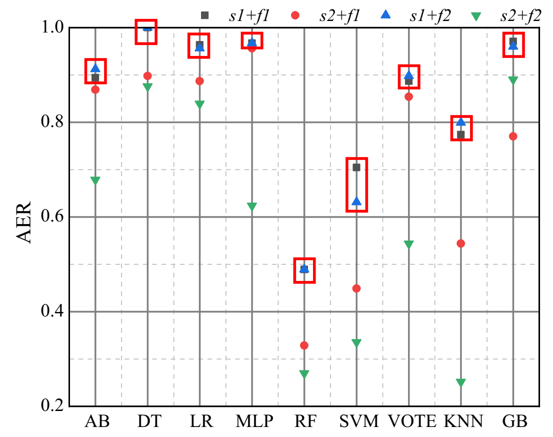

Fig. 7 The AER of different combination

The D-loss of LSGAN (D1 loss), the D-loss of MalGAN (D2 loss), the G-loss of LSGAN (G1 loss), and the G-loss of MalGAN (G2 loss) have reached convergence after the training of 20 epochs.

2. Different combinations of networks and loss functions in generating AME

We intend to demonstrate the effectiveness of the new network structure of LSGAN. Hu and Tan (2017) generated adversarial examples based on GAN, using Sigmoid Cross-Entropy (SCE) as a loss function and reducing the accuracy of almost all detectors. As a result, we set a Control Experiment (CE) to accomplish the structure comparison. As shown as Table 13, *s1* denotes the structure of LSGAN, *s2* is the structure of MalGAN, *f1* denotes the LS loss, and *f2* is the SCE loss. The combinations include LSGAN using *s1* and *f1* LS loss, CE1 using *s2* and *f1* LS loss, CE2 using *s1* and *f2* SEC loss, and MalGAN using *s2* and *f2* SEC loss. As the result of Table 13, the LSGAN network structure we proposed shows a better performance even though combines different loss functions. The LSGAN beats MalGAN under the 5 detectors. Furthermore, in the KNN and GB detectors, LSGAN combination shows closed result with *s1*+*f2* which is best in KNN and GB.

As shown in Fig. 7, the AERs of different combinations are computed. Using *s1* has better AER performance than using *s2*. Furthermore, the *s1*+*f1* combination has the best AER in 5 detectors, shown as a red rectangle in Fig. 7. It means that LSGAN possesses superior structures, and the LS loss function helps play our structure maximum performance in the other 5 detectors.

3. Comparision of recognition rate

In this subsection, we conduct 8 CEs in order to compare the performance of LSGAN-AT. We accomplish the state-of-the-art adversarial examples generating method PGD (Madry et al. 2018) and adversarial defense method TRADES (Zhang et al. 2019). Although classical PGD was proposed in 2018 ICLR and TRADES was proposed in 2019 ICML, extensive researches have verified the effectiveness of PGD-AT (PGD Adversarial Training) and TRADES in 2020 ICLR (Pang et al. 2021).

At first, we focus on three generative methods. We deploy 8 CEs using GM1/GM2/GM3 and RMD1/RMD2/RMD3. GM1 represents the Generating Model of LSGAN, GM2 represents the Generating Model of MalGAN, and GM3 represents the Generating Model of PGD. RMD1 represents AT module trained by LSGAN, RMD2 represents AT module trained by MalGAN, and RMD3 represents AT module trained by TRADES. Notice that, in this paper, we tend to illustrate the effectiveness of the brand-new network and proper activation function group of LSGAN. Thus, we only use the unique design of loss function and some optimizer parameters of PGD and TRADES.

We disclose parameters of CEs. MalGAN uses the same network structure and parameters with (Hu and Tan 2017). In the TRADES, we use the default parameters including SGD learning rate 0.01, momentum 0.5, and the robust loss coefficient 1.0. Similarly, in the PGD attack, we also use the default parameters including SGD learning rate 0.001, momentum 0.01, and uniform distribution epsilon 0.3. Note that, we all select the mean REC of 10 times experiments.

The results of REC of different combinations are shown in Table 14. We use GM1+RMD1 to represent the combination of LSGAN and RMD1, namely

Table 14 The recognition rate comparision of different combinations (%)

Combination	GM1 (LSGAN) as Attack Model			GM2 (MalGAN) as Attack Model			GM3 (PGD) as Attack Model		
	(Ours) GM1+RMD1	GM1+RMD2	GM1+RMD3	GM2+RMD1	GM2+RMD2	GM2+RMD3	GM3+RMD1	GM3+RMD2	GM3+RMD3
Train REC	65.90	38.66	52.59	56.49	51.09	52.25	71.27	65.76	68.42
Test REC	65.33	40.88	51.75	52.20	51.46	51.90	69.45	64.89	68.50

LSGAN-AT. Meanwhile, GM1 + RMD2, GM1 + RMD3, GM2 + RMD1, GM2 + RMD2, GM2 + RMD3, GM3 + RMD1, GM3 + RMD2, and GM3 + RMD3 have the same meaning. The REC of GM1 + RMD1 is 65.33% in the test set while the REC of GM1 + RMD2 is 40.88% and GM1 + RMD3 is 51.75%. It means that GM1 as attack model, using LSGAN to generate adversarial examples, is more effective than MalGAN and TRADES and RMD1 is more effective than RMD2 and TRADES in defense. At the same time, RMD1, namely LSGAN-AT, has the best REC testing by the same dataset when GM2 and GM3 as the attack model.

Like "LSGAN-AT transferability" section, our RMD1 has preferable transferability in each GM. It indicates RMD1 is available to recognize AME generated by GM2 and GM3 and RMD1 performs better than RMD2 and RMD3.

Conclusion

In this paper, we propose a LSGAN-AT approach including the LSGAN module and the AT module. In the LSGAN module, we deploy a well-designed union detector to fit Multi-Layer Perceptron (MLP) which has been selected by employing several experiments. After that, we utilize Least Square (LS) loss in the Generative Adversarial Network (GAN) in terms of a generator and a discriminator with brand-new network structures for adversarial training to generate Adversarial Malware Examples (AME). In the AT module, we develop a malware detector using Feed-Forward Neural Network (FFNN)-based new structure, trained by AME for generating Robust Malware Detector.

In the experimental section, we conduct numerous experiments to carry out LSGAN-AT evaluation, component selection analysis, LSGAN-AT transferability analysis, and model comparison analysis. The abundant experimental results indicate that the AME, generated by the LSGAN module, is effective to avoid the detection of detectors and has preferable transferability to attack other ML-based detectors. Furthermore, after trained by Adversarial Training (AT) module with AME and common dataset, RMD demonstrates the effectiveness in adversarial defense. In conclusion, LSGAN-AT can immensely enhance the robustness of malware detection.

Abbreviations

AB: AdaBoost; DT: Decision tree; DL: Deep learning; DNN: Deep neural network; FFNN: Feed-forward neural network; FGSM: Fast gradient sign method; GB: Gradient boosting decision tree; JSMA: Jacobian-based saliency method attack; KNN: K-nearest neighbor; LR: Logistic regression; MLP: Multi-layer perceptron; ML: Machine learning; NN: Neural network; SVM: Support vector machine; API: Application programming interface; AME: Adversarial malware example; AML: Adversarial machine learning; AT: Adversarial training; D: Discriminator; G: Generator; UD: Union detector; MD: Malware detector; RMD: Robust malware detector; GAN: Generative adversarial network; LSGAN: Least square generative adversarial network; MSE: Mean-square error; ReLU: Rectified linear units; LeakyReLU: Leaky rectified linear units.

List of symbols

M: The malware set; B: The benign set; M': The AME set generated by G; z: Slight perturbance; $L(G)$: The loss function of the generator; $L(D)$: The loss function of the discriminator; G_{θ_g}: The initialized parameter of G; D_{θ_g}: The initialized parameter of D; $G_{\theta_g}(x, z)$: AME generated by G; $D_{\theta_g}\left(G_{\theta_g}(x, z)\right)$: The probability value of AME discriminated by *D*; UD_{Benign}: A benign set classified by UD; $UD_{Malware}$: A malware set classified by UD; REC_{adv}: The recognition rate of AME; REC_{mal}: The recognition rate of malware.

Authors' contributions

Drafting the manuscript: JW and YW.; Revising the manuscript critically for important intellectual content: XC and RJR; experiments deployment: JW and JZ. All authors read and approved the final manuscript.

Authors' Information

Jianhua Wang he received the B.S. degree and M.S. degree in Software engineering from Taiyuan University of Technology in 2017 and 2020. He now pursues for his Ph.D. degree in Beijing Jiaotong University, major in Cyberspace Security. His research interests include adversarial machine learning and federated learning.

Dr. Xiaolin Chang is a Professor at School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include Cloud-Edge computing, network security, secure and dependable machine learning. She is a senior member of IEEE. She is the corresponding author of this paper.

Yixiang Wang he received his B.S. (2018) degree from Beijing Jiaotong University, China. He is a Ph.D. candidate at Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University. His research interests include adversarial examples, and security in machine learning.

Dr. Ricardo J. Rodríguez is an Associate Professor at the University of Zaragoza (Spain) since April 2021. His current research interests include the binary analysis of programs, especially applied in memory forensics and malware analysis.

Jianan Zhang she received the M.S. degree from Beijing Jiaotong University in 2021. Her research interests are mainly in adversarial example and security in machine learning.

Funding

The research of J. Wang, X. Chang, Y. Wang and J. Zhang was supported in part by Project supported by Chinese National Key Laboratory of Science and Technology on Information System Security and National Natural Science Foundation of China under Grant No. U1836105. The research of R. J. Rodríguez and X. Chang has been supported in part by the University of Zaragoza and the Fundación Ibercaja under Grant JIUZ-2020-TIC-08. The research of R. J. Rodríguez has also been supported in part by the University, Industry and Innovation Department of the Aragonese Government under *Programa de Proyectos Estratégicos de Grupos de Investigación* (DisCo research group, ref. T21-20R).

Declarations

Ethics approval and consent to participate

Not applicable.

Competing interests

No potential conflict of interest was reported by the authors.

Author details

[1]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, Beijing 100044, China. [2]Department of Computer Science and Systems Engineering, University of Zaragoza, Calle María de Luna 1, Zaragoza 50018, Spain.

Received: 30 July 2021 Accepted: 29 October 2021
Published online: 02 December 2021

References

Allix K, Bissyandé TF, Klein J, Le Traon Y (2016) AndroZoo: collecting millions of Android apps for the research community. In: Proceedings of the 13th international conference on mining software repositories, Austin Texas, May 2016, pp 468–471. https://doi.org/10.1145/2901739.2903508

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In Proceedings of the 34th international conference on machine learning, Jul. 2017, pp 214–223. Accessed: Sep. 24, 2021. https://proceedings.mlr.press/v70/arjovsky17a.html

Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Siemens C (2014) Drebin: effective and explainable detection of android malware in your pocket. Ndss 14:23–26

Berlin K, Slater D, Saxe J (2015) Malicious behavior detection using windows audit logs. In: Proceedings of the 8th ACM workshop on artificial intelligence and security, pp 35–44

Biggio B, Roli F (2018) Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recognit 84:317–331

Chen S et al (2018) Automated poisoning attacks and defenses in malware detection systems: an adversarial machine learning approach. Comput Secur 73:326–344

Chen L, Ye Y, Bourlai T (2017) Adversarial machine learning in malware detection: arms race between evasion attack and defense. In: 2017 European intelligence and security informatics conference (EISIC), 2017, pp 99–106

Chocolatey—The package manager for Windows, Chocolatey Software. https://chocolatey.org/ (accessed Jul. 21, 2021)

Christodorescu M, Jha S, Seshia SA, Song D, Bryant RE (2005) Semantics-aware malware detection. In: 2005 IEEE symposium on security and privacy (S&P'05), 2005, pp 32–46

Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA (2018) Generative adversarial networks: an overview. IEEE Signal Process Mag 35(1):53–65

"Cuckoo Sandbox—Automated Malware Analysis." https://cuckoosandbox.org/. Accessed April 23, 2021

Damodaran A, Troia FD, Visaggio CA, Austin TH, Stamp M (2017) A comparison of static, dynamic, and hybrid analysis for malware detection. J Comput Virol Hacking Tech 13(1):1–12. https://doi.org/10.1007/s11416-015-0261-z

Gonog L, Zhou Y (2019) A review: generative adversarial networks. In: 2019 14th IEEE conference on industrial electronics and applications (ICIEA), 2019, pp 505–510

Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P (2016) Adversarial perturbations against deep neural networks for malware classification. arXiv: http://arxiv.org/abs/1606.04435, 2016

Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P (2017) Adversarial examples for malware detection. In European symposium on research in computer security, pp 62–79

Hu W, Tan Y (2017) Generating adversarial malware examples for black-box attacks based on GAN," ArXiv170205983 Cs, Feb. 2017, Accessed: Apr. 23, 2021. http://arxiv.org/abs/1702.05983

Huang L, Joseph AD, Nelson B, Rubinstein BI, Tygar JD (2011) Adversarial machine learning. In: Proceedings of the 4th ACM workshop on security and artificial intelligence, 2011, pp 43–58

Huang S, Papernot N, Goodfellow I, Duan Y, Abbeel P (2017) Adversarial attacks on neural network policies. ArXiv170202284 Cs Stat, Feb. 2017, Accessed: Jul. 14, 2021. [Online]. Available: http://arxiv.org/abs/1702.02284

Khoda ME, Imam T, Kamruzzaman J, Gondal I, Rahman A (2019) Robust malware defense in industrial IoT applications using machine learning with selective adversarial samples. IEEE Trans Ind Appl 56(4):4415–4424

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. ArXiv Prepr. http://arxiv.org/abs/1412.6980

Li D, Li Q (2020) Adversarial deep ensemble: evasion attacks and defenses for malware detection. IEEE Trans Inf Forensics Secur 15:3886–3900. https://doi.org/10.1109/TIFS.2020.3003571

Lucas K, Sharif M, Bauer L, Reiter MK, Shintre S (2021) Malware Makeover: breaking ML-based static analysis by modifying executable bytes. In: Proceedings of the 2021 ACM Asia conference on computer and communications security, New York, NY, USA, May 2021, pp. 744–758. https://doi.org/10.1145/3433210.3453086

Lugmayr A, Danelljan M, Van Gool L, Timofte R (2020) Srflow: learning the super-resolution space with normalizing flow. In: European conference on computer vision, 2020, pp 715–732

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. https://openreview.net/forum?id=rJzIBfZAb

Malware Statistics & Trends Report | AV-TEST." https://www.av-test.org/en/statistics/malware/. Accessed Jul. 14, 2021

Mao X, Li Q, Xie H, Lau RYK, Wang Z, Paul Smolley S (2017) Least squares generative adversarial networks, 2017, pp. 2794–2802. Accessed: Apr. 23, 2021. https://openaccess.thecvf.com/content_iccv_2017/html/Mao_Least_Squares_Generative_ICCV_2017_paper.html

Microsoft Malware Classification Challenge (BIG 2015). https://kaggle.com/c/malware-classification (accessed Jul. 21, 2021)

Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines

Nappa A, Rafique MZ, Caballero J (2015) The MALICIA dataset: identification and analysis of drive-by download operations. Int J Inf Secur 14(1):15–33

Nash JF (2016) 8. Two-person cooperative games. Princeton University Press, Princeton

Pan Z et al (2020) Loss functions of generative adversarial networks (GANs): opportunities and challenges. IEEE Trans Emerg Top Comput Intell 4(4):500–522. https://doi.org/10.1109/TETCI.2020.2991774

Pang T, Yang X, Dong Y, Su H, Zhu J (2021) Bag of tricks for adversarial training. https://openreview.net/forum?id=Xb8xvrtB8Ce

Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy (EuroS&P), 2016, pp 372–387

Pedregosa F et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Sewak M, Sahay SK, Rathore H (2020) DOOM: a novel adversarial-DRL-based op-code level metamorphic malware obfuscator for the enhancement of IDS. In: Adjunct proceedings of the 2020 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2020 ACM international symposium on wearable computers, 2020, pp 131–134

Suciu O, Coull SE, Johns J (2019) Exploring adversarial examples in malware detection. In: 2019 IEEE security and privacy workshops (SPW), 2019, pp 8–14

Szegedy C et al (2014) Intriguing properties of neural networks. ArXiv13126199 Cs, Feb. 2014, Accessed: Apr. 23, 2021. [Online]. http://arxiv.org/abs/1312.6199

VirusTotal. https://www.virustotal.com/gui/ (accessed Jul. 21, 2021)

VirusShare.com." https://virusshare.com/research (accessed Apr. 23, 2021)

Wang X, Li J, Kuang X, Tan Y, Li J (2019a) The security of machine learning in an adversarial setting: a survey. J Parallel Distrib Comput 130:12–23

Wang D, Dong L, Wang R, Yan D, Wang J (2020) Targeted speech adversarial example generation with generative adversarial network. IEEE Access 8:124503–124513

Wang Y, Liu J, Chang Z (2019) Assessing transferability of adversarial examples against malware detection classifiers. In: Proceedings of the 16th ACM international conference on computing frontiers, 2019, pp 211–214

Wang Q et al. (2017) Adversary resistant deep neural networks with an application to malware detection. In: Proceedings of the 23rd ACM sigkdd international conference on knowledge discovery and data mining, 2017, pp 1145–1153

Xiao C, Li B, Zhu J, He W, Liu M, Song D (2018) Generating adversarial examples with adversarial networks. In: Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18, Jul. 2018, pp 3905–3911. https://doi.org/10.24963/ijcai.2018/543

Xiao C, Li B, Zhu J-Y, He W, Liu M, Song D (2016) Generating adversarial examples with adversarial networks. Accessed: Apr. 29, 2021. http://arxiv.org/abs/1801.02610

Yuan X, He P, Zhu Q, Li X (2019) Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017

Yuan Z, Lu Y, Wang Z, Xue Y (2014) Droid-Sec: deep learning in android malware detection. In: Proceedings of the 2014 ACM conference on

SIGCOMM, New York, NY, USA, Aug. 2014, pp. 371–372. https://doi.org/10.1145/2619239.2631434
Yuan J, Zhou S, Lin L, Wang F, Cui J (2020) Black-box adversarial attacks against deep learning based malware binaries detection with GAN, ECAI 2020, pp 2536–2542. https://doi.org/10.3233/FAIA200388
Zhang H, Yu Y, Jiao J, Xing E, El Ghaoui L, Jordan M (2019) Theoretically principled trade-off between robustness and accuracy. In: International conference on machine learning, pp 7472–7482
Zhang Y, Li H, Zheng Y, Yao S, Jiang J (2021) Enhanced DNNs for malware classification with GAN-based adversarial training. J Comput Virol Hacking Tech 17(2):153–163. https://doi.org/10.1007/s11416-021-00378-y
Zhao J, Mathieu M, LeCun Y (2017) Energy-based generative adversarial networks: 5th international conference on learning representations, ICLR 2017. Accessed: Sep. 24, 2021. http://www.scopus.com/inward/record.url?scp=85087518435&partnerID=8YFLogxK
Zhou Y, Jiang X (2012) Dissecting android malware: characterization and evolution. In: 2012 IEEE symposium on security and privacy, May 2012, pp 95–109. https://doi.org/10.1109/SP.2012.16

Publisher's Note